

Oggetti distribuiti in Python

PyRO

- Python mette a disposizione un modulo software per l'istanziamento di oggetti remoti (in stile Java RMI)
- Il modulo in questione è PyRO (Python Remote Objects)
- Installazione (sistemi Debian GNU/Linux)
`sudo apt-get install pyro`

Caratteristiche

- **PyRO implementa un protocollo di comunicazione basato sul meccanismo di serializzazione pickle**
- **Tale protocollo si fonda sul TCP**
- **Server e client possono scambiarsi i ruoli (paradigma P2P, oltre che il classico client-server)**
- **Quasi trasparente all'utente**
- **Altamente configurabile**

Principi di uso

- **PyRO usa un proprio object name server (NS) per individuare gli oggetti remoti**
 - **Gli oggetti remoti devono essere registrati presso il NS**
 - **I client interrogano il NS al fine di individuare un oggetto ed ottenere un URI (Uniform Resource Identifier, simile a quelli usati nel WWW)**
 - **L'interrogazione avviene in broadcast**
- **I client usano proxy per smistare le invocazioni dei metodi gli oggetti remoti**
 - **Proxy statici**
 - **Proxy dinamici**
 - **Proxy dinamici con supporto agli attributi**

Un semplice esempio

- **Illustriamo il funzionamento di PyRO mediante un esempio**
 - **tst.py: codice che effettua i calcoli (eseguito in remoto)**
 - **server.py: server che esporta le funzionalità di tst.py**
 - **client.py: client che invoca i metodi di tst.py in locale e li consegna a server.py**
- **Una collezione esaustiva di esempi può essere ottenuta installando il pacchetto pyro-examples**

tst.py

```
class testclass:
```

```
    def mul(s, arg1, arg2): return arg1*arg2
```

```
    def add(s, arg1, arg2): return arg1+arg2
```

```
    def sub(s, arg1, arg2): return arg1-arg2
```

```
    def div(s, arg1, arg2): return arg1/arg2
```

```
    def error(s):
```

```
        x=foo()
```

```
        x.crash()
```

```
class foo:
```

```
    def crash(s):
```

```
        s.crash2('going down...')
```

```
    def crash2(s, arg):
```

```
        # this statement will crash on purpose:
```

```
        x=arg/2
```

server.py

```
import Pyro.core
import Pyro.naming
from Pyro.errors import NamingError
import tst

class testclass(Pyro.core.ObjBase, tst.testclass):
    def __init__(self):
        Pyro.core.ObjBase.__init__(self)

Pyro.core.initServer()
ns=Pyro.naming.NameServerLocator().getNS()
daemon=Pyro.core.Daemon()
daemon.useNameServer(ns)
uri=daemon.connect(testclass(),":test.simple")

print "Server is ready."
daemon.requestLoop()
```

client.py

```
import Pyro.util  
import Pyro.core
```

```
Pyro.core.initClient()
```

```
test = Pyro.core.getProxyForURI("PYRONAME://:test.simple")
```

```
print test.mul(111,9)  
print test.add(100,222)  
print test.sub(222,100)  
print test.div(2.0,9.0)  
print test.mul('.',10)  
print test.add('String1','String2')
```

```
print '*** invoking server method that crashes ***'  
print test.error()
```


Invocazione del meccanismo

- **Useremo tre shell (shell1, shell2, shell3)**
 - **Possono eseguire su tre computer diversi**
- **Sulla shell1, facciamo partire il name server pyro-ns**
- **Sulla shell2, facciamo partire il server Pyro
./server.py**
- **Sulla shell3, facciamo partire il client Pyro
./client.py**

Uso in rete locale

- **Attenzione! Sia il name server che l'object server ascoltano sull'interfaccia di rete di default dell'host**
- **In parecchie distribuzioni GNU/Linux, tale interfaccia corrisponde all'indirizzo localhost (127.0.0.1 o 127.0.1.1)**
- **Sotto tali condizioni, PyRO non funziona in rete locale!**

Uso in rete locale

- **Passi da compiere per far funzionare PyRO in rete locale**
 - **Passo 1: assicurarsi che il nome dell'host si risolva nell'IP dell'interfaccia di rete (non in localhost)**
 - **Se necessario, modificare il file /etc/hosts**
- host nome_dell_host
155.185.30.2 ----> OK
127.0.0.1 ---> ERRORE

Uso in rete locale

- **Passi da compiere per far funzionare PyRO in rete locale**
- **Passo 2: assicurarsi che il server ascolti sull'interfaccia di rete, e non su localhost**
`daemon=Pyro.core.Daemon(host="nome_dell_host")`
- **Passo 3: assicurarsi che il name server ascolti sull'interfaccia di rete, e non su localhost**
`pyro-ns -n nome_host_name_server`